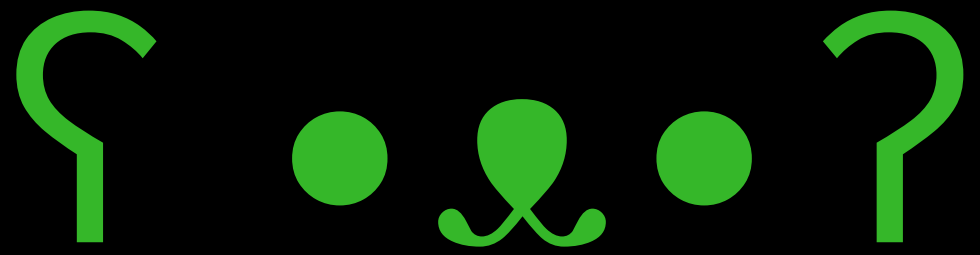


b34r5hell

Reverse Engineering

Agenda

> What is Reverse Engineering?
> How does it apply to CTFs?
> objdump
> GDB
> Decompilers

What is Reverse Engineering?

- **Understanding how something you didn't make works by deductive reasoning and analysis**
  - Often with the goal of modifying that thing or gaining valuable information to achieve some goal

- **Example Situations that utilize RE**
  - Creating an emulator for a video game console
  - Modding a video game (without modding tools specific to that game)
  - Finding a vulnerability in proprietary software
  - Bypassing security checks on a piece of hardware/software

How does it apply to CTFs?

- **Usually given an executable to download, and asked to figure something out about it**
  - E.g. what is the correct user input that will give you the flag?
  - You also see non-compiled languages like Python
- **Sometimes overlap with Binary Exploitation and/or “pwn” challenges**
  - Exploits are often found using reverse engineering techniques
- **Challenges can be made more complex by using unconventional programming techniques or obscure libraries**
  - Often tough to figure out “where to look”, challenge makers put a lot of red-herrings to annoy us

objdump

- **Command line disassembler**
  - Can convert executable machine code into assembly code
  - Analyzes executables and object files

```
objdump -d ./file | less
```

GNU Debugger(GDB)

- **Industry-standard debugger for Linux executables**
  - Set breakpoints in the assembly
  - Print values in registers/memory at any point

- **Can “cheat” by manually running specific sections of code, or bypassing conditionals**
  - Easier challenges might be solvable like this without having to do any actual analysis
  - Don’t get too comfortable with this: challenge makers know all our tricks

- **Useful to combine with a decompiler so that you can figure out where in the “code” you are currently at**

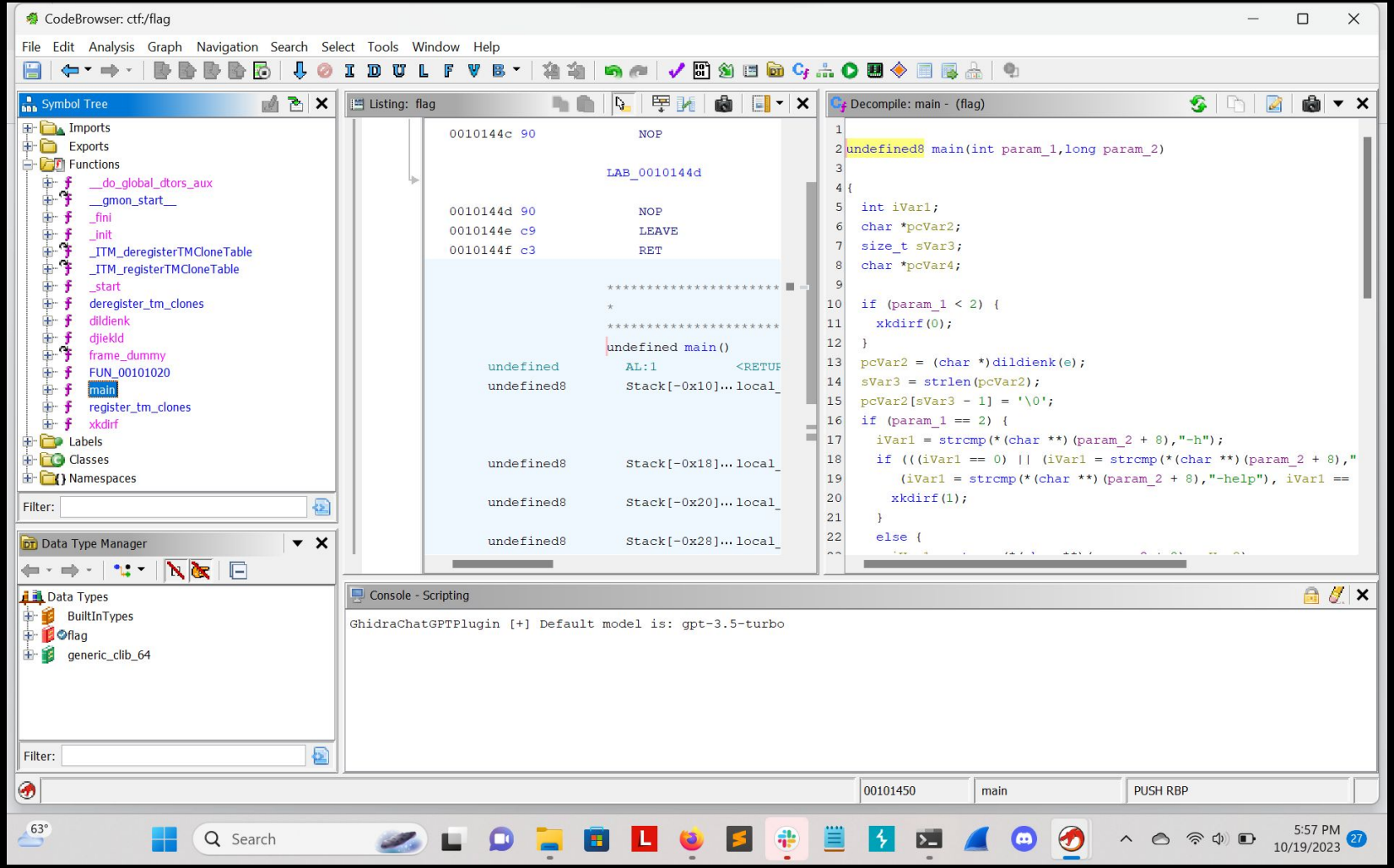
Decompilers

- **By themselves, an executable will only give you the raw machine code**
  - Possible to solve a challenge only using tools for parsing executables and debugging assembly
  - But often impractical and takes way too long

- **Luckily, there are tools that can take a compiled executable and “decompile” it into C/C++**
  - Will never be exactly correct, decompilers make a good guess
  - Will still lack important information like variable names
  - Even if functionally similar, will often look completely different than what the original source looked like

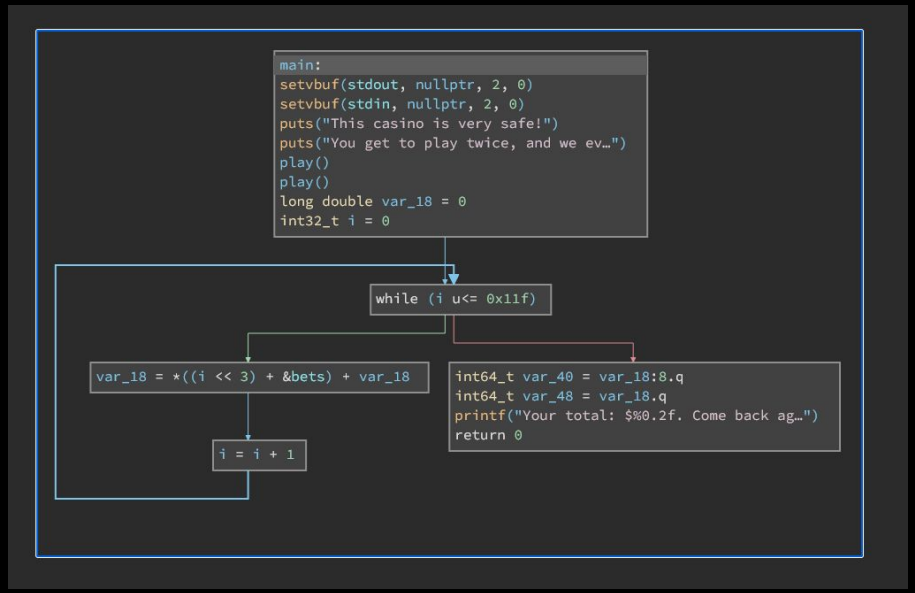
Ghidra

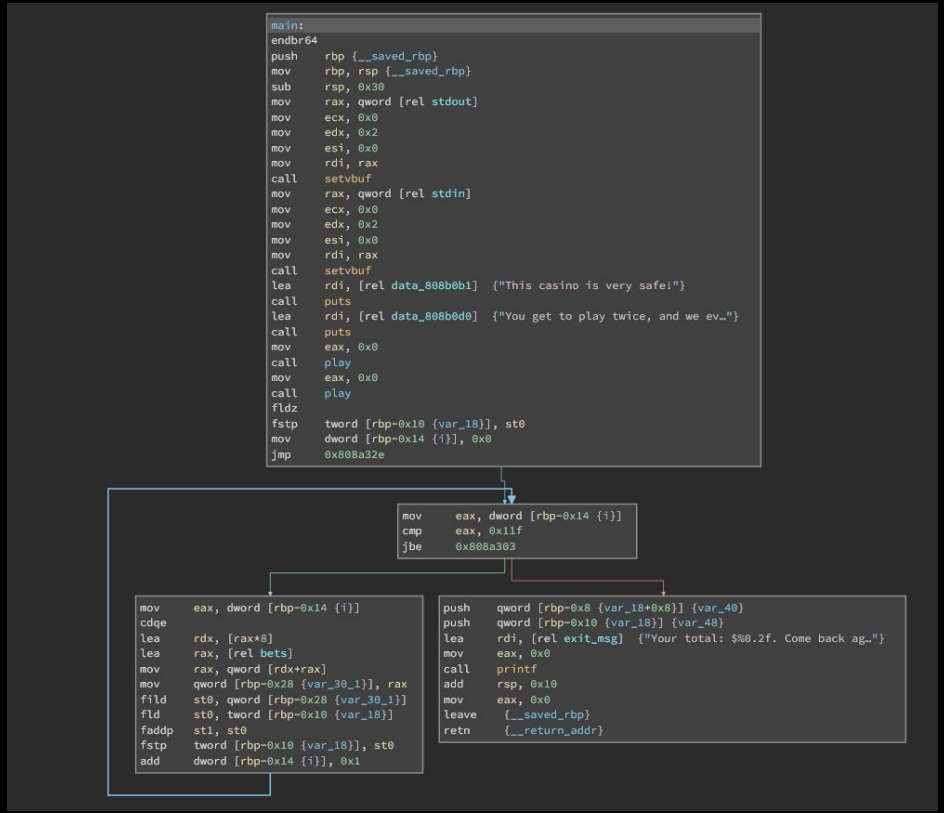
- Made by the NSA and released in 2019
  - Has existed and been used for much longer within government agencies
- Written in Java, so it works on all operating systems with a valid Java version
- Open-Source, available on GitHub: ghidra
- Can also do cool plugins or scripts
- The decompiler we tend to use


CodeBrowser: ctf:/flag
File Edit Analysis Graph Navigation Search Select Tools Window Help
Symbol Tree
Imports
Exports
Functions
__do_global_dtors_aux
__gmon_start__
_fini
_init
_ITM_deregisterTMCloneTable
_ITM_registerTMCloneTable
_start
deregister_tm_clones
dildienk
djiekld
frame_dummy
FUN_00101020
main
register_tm_clones
xkdirf
Labels
Classes
Namespaces
Filter:
Data Type Manager
Data Types
BuiltInTypes
flag
generic_clib_64
Filter:
Listing: flag
0010144c 90 NOP
LAB_0010144d
0010144d 90 NOP
0010144e c9 LEAVE
0010144f c3 RET
undefined main()
undefined AL:1 <RETUR
undefined8 Stack[-0x10]... local_
undefined8 Stack[-0x18]... local_
undefined8 Stack[-0x20]... local_
undefined8 Stack[-0x28]... local_
Decompile: main - (flag)
undefined8 main(int param_1,long param_2)
{
int iVar1;
char *pcVar2;
size_t sVar3;
char *pcVar4;
if (param_1 < 2) {
xkdirf(0);
}
pcVar2 = (char *)dildienk(e);
sVar3 = strlen(pcVar2);
pcVar2[sVar3 - 1] = '\0';
if (param_1 == 2) {
iVar1 = strcmp(*(char **)(param_2 + 8),"-h");
if (((iVar1 == 0) || (iVar1 = strcmp(*(char **)(param_2 + 8),"
(iVar1 = strcmp(*(char **)(param_2 + 8),"-help"), iVar1 ==
xkdirf(1);
}
else {
Console - Scripting
GhidraChatGPTPlugin [+] Default model is: gpt-3.5-turbo
00101450
main
PUSH RBP
Search
5:57 PM
10/19/2023

Binary Ninja

- Both an online and local version
  - The local tool costs money
  - Online version is free: https://binary.ninja/
    - Go to cloud to use the online tool
- Provides similar functionality to Ghidra
  - Has a nice GUI and provides cleaner control flow graphs (CFG), IMO
- Provides a lot of intermediate representations (IR)


main:
setvbuf(stdout, nullptr, 2, 0)
setvbuf(stdin, nullptr, 2, 0)
puts("This casino is very safe!")
puts("You get to play twice, and we ev…")
play()
play()
long double var_18 = 0
int32_t i = 0
while (i u<= 0x11f)
var_18 = *((i << 3) + &bets) + var_18
i = i + 1
int64_t var_40 = var_18:8.q
int64_t var_48 = var_18.q
printf("Your total: $%0.2f. Come back ag…")
return 0


main:
endbr64
push rbp {__saved_rbp}
mov rbp, rsp {__saved_rbp}
sub rsp, 0x30
mov rax, qword [rel stdout]
mov ecx, 0x0
mov edx, 0x2
mov esi, 0x0
mov rdi, rax
call setvbuf
mov rax, qword [rel stdin]
mov ecx, 0x0
mov edx, 0x2
mov esi, 0x0
mov rdi, rax
call setvbuf
lea rdi, [rel data_808b0b1] {"This casino is very safe!"}
call puts
lea rdi, [rel data_808b0d0] {"You get to play twice, and we ev…"}
call puts
mov eax, 0x0
call play
mov eax, 0x0
call play
fldz
fstp tword [rbp-0x10 {var_18}], st0
mov dword [rbp-0x14 {i}], 0x0
jmp 0x808a32e
mov eax, dword [rbp-0x14 {i}]
cmp eax, 0x11f
jbe 0x808a303
mov eax, dword [rbp-0x14 {i}]
cdqe
lea rdx, [rax*8]
lea rax, [rel bets]
mov rax, qword [rdx+rax]
mov qword [rbp-0x28 {var_30_1}], rax
fild st0, qword [rbp-0x28 {var_30_1}]
fld st0, tword [rbp-0x10 {var_18}]
faddp st1, st0
fstp tword [rbp-0x10 {var_18}], st0
add dword [rbp-0x14 {i}], 0x1
push qword [rbp-0x8 {var_18+0x8}] {var_40}
push qword [rbp-0x10 {var_18}] {var_48}
lea rdi, [rel exit_msg] {"Your total: $%0.2f. Come back ag…"}
mov eax, 0x0
call printf
add rsp, 0x10
mov eax, 0x0
leave {__saved_rbp}
retn {__return_addr}

IDA

- Commonly used in industry for reverse engineering
- SUPER expensive (~$5000 for the base)
- Has a limited free version called IDA Free

Tasks

- Download Ghidra
- Complete the reverse engineering module in the dojo